

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2001-005725

(43)Date of publication of application : 12.01.2001

(51)Int.CI.

G06F 12/12

(21)Application number : 11-173656

(71)Applicant : HITACHI LTD

(22)Date of filing : 21.06.1999

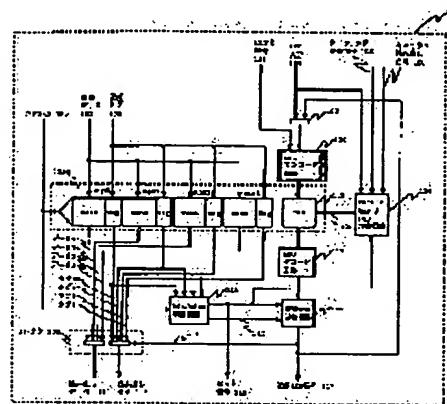
(72)Inventor :
NISHIMOTO SATORU
IIZUKA TAKAYOSHI
KAMATA EIKI

(54) CACHE STORAGE DEVICE

(57)Abstract:

PROBLEM TO BE SOLVED: To provide a cache storage device which decreases a frequency of miscaching due to a block conflict as a set associative type cache storage device.

SOLUTION: The set associative type cache memory device equipped with a memory cell part which holds data and tags, an LRU memory part representing reference history information on cache blocks, and a circuit which makes a hit/miss decision has an instruction for changing the LRU of a corresponding cache block to the block which was referred to in the remote past. When the instruction is executed, an LRU encoding circuit 110 changes the LRU of the corresponding block to the block which was referred to in the remote past.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

Japanese Laid-Open Patent Application No. 2001-5725 Publication

(Lines 31 to 47)

...When the change-to-LRU store instruction is executed, first, the cache is accessed using the given address. When a cache miss occurs, data is written into the main memory and the process is ended, as a store-through is assumed in the present embodiment. In the case of a cache hit, the ROW in which the hit occurs is ouputted from the selection ROW signal 117. Subsequently, when store data is written into a cache, the selection ROW signal 117 is inputted to the LRU encode circuit 110; the LRU encode circuit 110 creates a value to be written into the LRU array 111; and along with updating the data of the cache, the value of the LRU is also updated. Moreover, here, the LRU encode circuit 110 operates according to the table 303 in FIG. 3, in the same manner as during the LRU updating in the aforementioned change-to-LRU store instruction. With this, the LRU of the block in which data was written into according to the change-to-LRU store instruction becomes the block accessed in the most distant past.

FIG. 3

<Replace row determination: decoding table>

301

NEVER ROW#	(0)	(1)	(2)	(3)	(4)	(5)
row0	0			0	0	
row1	1	0				0
row2			0	1		1
row3		1	1		1	

<Normal memory referenced LRU updating: encoding table>

302

NEVER ROW#	(0)	(1)	(2)	(3)	(4)	(5)
row0	1			1	1	
row1	0	1				1
row2			1	0		0
row3		0	0		0	

<Change-to-LRU memory referenced LRU updating: encoding table>

303

NEVER ROW#	(0)	(1)	(2)	(3)	(4)	(5)
row0	0			0	0	
row1	1	0				0
row2			0	1		1
row3		1	1		1	

(19)日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開2001-5725

(P2001-5725A)

(43)公開日 平成13年1月12日 (2001.1.12)

(51)Int.Cl.
G 0 6 F 12/12

識別記号

F I
G 0 6 F 12/12

テ-7J-ト*(参考)
D 5 B 0 0 5
A
F

審査請求 未請求 請求項の数1 O L (全 11 頁)

(21)出願番号 特願平11-173656

(22)出願日 平成11年6月21日(1999.6.21)

(71)出願人 000005108

株式会社日立製作所
東京都千代田区神田駿河台四丁目6番地

(72)発明者 西本 哲
神奈川県川崎市麻生区王禅寺1099番地 株
式会社日立製作所システム開発研究所内

(72)発明者 飯塚 孝好
神奈川県川崎市麻生区王禅寺1099番地 株
式会社日立製作所システム開発研究所内

(74)代理人 100075096
弁理士 作田 康夫

最終頁に続く

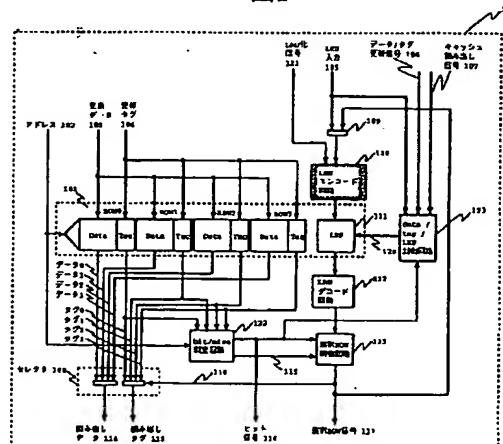
(54)【発明の名称】 キャッシュ記憶装置

(57)【要約】

【課題】 セットアソシアティブ方式のキャッシュ記憶装置において、ブロック競合によるキャッシュミス回数を低減するキャッシュ記憶装置を提供すること。

【解決手段】 データとタグを保持するメモリセル部と、キャッシュブロックの参照履歴情報を表すLRUメモリ部と、ヒット/ミスを判定する回路を具備するセットアソシアティブ方式のキャッシュメモリ装置において、該当キャッシュブロックのLRUを最も遠い昔に参照したブロックに変更する命令を備える。前記命令の実行時に、LRUエンコード回路110において該当ブロックのLRUを最も遠い昔に参照したブロックに変更する。

図1



【特許請求の範囲】

【請求項1】キャッシュブロックの追い出しアルゴリズムにLRUを使用したセットアソシアティブ方式のキャッシュ記憶装置を備えるプロセッサにおいて、キャッシュキャッシュブロックが最も遠い昔にアクセスされたブロックとなるように、キャッシュブロックの参照履歴情報を更新する(以後"LRU化する"と呼ぶ)命令(以後、"LRU化命令"と呼ぶ)を備えることを特徴とするプロセッサ。

【発明の詳細な説明】

【発明の属する技術分野】本発明はプロセッサにおけるキャッシュ記憶装置に係わり、特に、将来使用される可能性の高いデータのキャッシュからの追い出しを抑止することができるキャッシュ記憶装置に関する。

【従来の技術】キャッシュ記憶装置については、富田眞治、村上和彰、新實治男 訳：“コンピュータアーキテクチャ：設計・実現・評価の定量的アプローチ”：P403～P428：日経BP、において詳しく述べられている。コンピュータプログラムの主記憶参照には局所性がある。この性質を利用して、頻繁にアクセスされる主記憶データをキャッシュ記憶装置と呼ばれる高速で小容量のメモリにコピーし、主記憶へのアクセスをこのキャッシュ記憶装置に対して行なうことによって、メモリアクセスを高速化させることができて行なわれている。キャッシュ記憶装置と主記憶の間のデータの管理は、ブロックと呼ぶ適当な大きさを単位として行なわれる。ブロックの大きさをブロックサイズと呼ぶ。キャッシュ記憶装置には複数のブロックが格納されており、例えばキャッシュサイズが128Kバイト、ブロックサイズが128バイトの場合、キャッシュ記憶装置には1024個のブロックが格納される。キャッシュに格納されたデータはデータアレイと呼ぶメモリに保持される。またデータと同様にキャッシュが主記憶上のどのブロックを保持するかを識別するために、ブロックのアドレスがアドレスアレイと呼ばれるメモリに保持される。プロセッサが参照するデータがキャッシュ上にあるかどうかは、アドレスアレイと、命令が参照するアドレスを比較することで判定する。主記憶上のブロックをキャッシュ上のどこに配置するかによって、キャッシュの構成は3つに分類できる。主記憶上の各ブロックのキャッシュ上での位置がそのアドレスによって一意に決められている場合をダイレクトマップ方式と呼び、主記憶上の各ブロックをキャッシュ上の任意の場所に配置して良い場合をフルアソシアティブ方式と呼び、主記憶上のブロックをキャッシュ上のある決められた範囲内の中だけに配置することができる場合を、セットアソシアティブ方式と呼ぶ。一般にフルアソシアティブ方式の場合、キャッシュ上にデータが存在するかどうかを判定するためには、命令が参照するアドレスと、キャッシュに格納されている全てのブロックのアドレスを比較しなければならず、ハードウェア量

の点から現実的ではない。そのためブロックのマッピングには、従来からダイレクトマップ方式と、セットアソシアティブ方式が使用されている。セットアソシアティブ方式の概要を図4に示す。セットアソシアティブ方式では、キャッシュメモリをN行M列のブロックに分割する。各ブロックにはデータとブロックのアドレスが格納されている。ここでキャッシュの各行をカラムと呼び、各列をロー(ROW)と呼ぶ。図4は、キャッシュサイズが128Kバイト、ブロックサイズが128バイトの10 4 ウエイセットアソシアティブのキャッシュを表している。この例では、各カラムは4個のブロックから構成され、キャッシュは256個のカラムから構成される。メモリアクセスが図4の401、402、403に示す40ビットのアドレスで行なわれると仮定すると、403の7ビットはブロック内のオフセットを表し、402の8ビットは、キャッシュのカラム番号を表し、401の25ビットは、同一カラムに格納される可能性のある主記憶上のブロックのインデックスを表す。したがって、アドレスアレイには401に示すブロックのインデックスが格納される。メモリ参照命令が実行されると、まず402で示されるカラムアドレスによって、キャッシュがアクセスされる。ここでカラムアドレスがiならば、キャッシュメモリのカラムi上の4個のブロックがアクセスされる。次に上記で読み出された4個のブロックのアドレスと、401で与えられるブロックのインデックスの一致を検査する。4個のいずれかのブロックとアドレスが一致していれば、キャッシュにヒットしたといい、参照するデータがキャッシュ上に存在することが分かる。4個のどのブロックとも一致しなければ、キャッシュミスしたといい、参照するデータはキャッシュ上に存在しないことが分かる。この場合、主記憶からブロックをキャッシュ上にコピーする。セットアソシアティブ方式では、主記憶からフェッチしたブロックを格納するキャッシュ上の位置の決定において、カラム番号はフェッチしたブロックのアドレスによって一意に決まるが、カラム内のどのROWにブロックを登録するかは、なんらかの方法で決定しなければならない。該当カラム内に有効でないROWが存在すれば、そのROWを置き換える対象とできるが、全てのROWが有効である場合は、カラム内のいずれかのROWを追い出して、その後、該当ROWに主記憶からのブロックをコピーする必要がある。上記の追い出しが発生する場合の、追い出し対象となるROWを決定する方法として、一般にLRU(Last Recently Used)アルゴリズムが使用されている。これは、最も遠い過去に参照されたブロックを追い出し対象とする方式である。図4におけるLRU407は、カラム内の4個のブロックの参照履歴情報を格納するためのメモリであり、カラム毎に対応するLRU情報を保持している。このLRU情報を用いて、50 カラム内の4個のブロックの中で最も遠い昔に参照され

たブロックを検出する。

【発明が解決しようとする課題】一般にプログラムのメモリアクセスパターンを解析すると、次のような特徴が現れる場合が非常に多い。

(1) あるアドレスのデータに対するアクセスは比較的短い時間内に再発する。

(2) ある一定時間内にアクセスされるデータは比較的近いアドレスに分布する。前者は「時間的局所性」、後者は「空間的局所性」と呼ばれる。セットアソシアティブ方式において、追い出しアルゴリズムにLRUを用いた場合、最も最近アクセスされたブロックは追い出されないので、メモリアクセスパターンに、"空間的局所性はあるが、時間的局所性がない"という特徴を持つ大きな配列をアクセスする場合、キャッシュ内のブロックが上記配列によって全て上書きされ、スタック上のデータなどの時間的局所性があるデータが追い出され、この追い出された時間的局所性のあるデータへの参照が再び発生した時にキャッシュミスが発生する場合がある。図5は、上記の問題が発生するプログラム例である。この例では、キャッシュサイズが128Kバイト、ブロックサイズが128バイトの4ウェイセットアソシアティブのキャッシュを仮定する。501はソープrogramであり、配列の各要素が8バイトである100×16000の配列Sを100×16000の配列Tにコピーする。内側ループ504で16000個の要素のコピーを行ない、外側ループ502を100回繰り返すことで、全要素のコピーを行なう。また外側ループの各繰り返しにおいて、外側ループ502内には、内側ループ504の実行が始まる前にスタック上の局所変数へのアクセス503があると仮定する。また配列S、Tは128バイト境界に配置されると仮定する。501のプログラムにおいて、配列S、Tに対するメモリ参照は、ある一定時間内に比較的近いアドレスに分布するが、同一アドレスに対する参照は一度しか発生しない。したがって、配列S、Tのメモリアクセスパターンには、空間的局所性はあるが、時間的局所性はないと考えられる。一方、局所変数の使用503で参照されるスタック上のデータは、外側ループ502の繰り返し毎に同一アドレスが参照されるので、時間的局所性があると考えられる。510は501の内側ループを疑似命令列で書き換えたものである。510の内側ループ513は、配列Sの各要素をロードして、配列Tの各要素にストアする疑似命令列になる。内側ループ513は16000回繰り返され、かつ配列Sの要素は8バイトなので、外側ループ511の1回の繰り返しで、キャッシュサイズと同じ128Kバイトメモリをシーケンシャルにアクセスする。このプログラムを実行すると、まず外側ループ511の最初の繰り返しで、局所変数の使用512によって、スタックのデータを格納するブロックがキャッシュ上にコピーされる。その後、内側ループ513の実行によって、128

Kバイトのメモリがアクセスされるため、スタックのデータを格納するキャッシュ上のブロックはキャッシュから追い出され、キャッシュ上のデータは全て配列Sによって上書きされてしまう。その後、外側ループ511の次の繰り返しの局所変数の使用512において、再びスタックが参照されたとき、キャッシュミスが発生する。スタック上のデータには時間的局所性があるため、この例では外側ループ511の繰り返し毎に、局所変数の使用512でキャッシュミスが発生する。このような、時間的局所性のあるブロックが、空間的局所性はあるが時間的局所性のないブロックによってキャッシュから追い出されるという問題を解決するために、特開平7-281957号で示されているLRUのロック方式を使用することができる。上記特許によれば、再利用される可能性のあるデータの最初の参照でLRUをロックし、該当データの最後の参照でLRUのロックを解除する。この方法を、前記のプログラム例に適用すると、スタックの最初の参照でLRUをロックし、関数を抜けるときにLRUのロックを解除することになる。これにより、スタック

20 のデータを保持するブロックはキャッシュ上で常に最新のままになるので、配列Sの参照によってスタックデータが追い出されることなくなる。しかしながら、LRUのロック方式ではプロセスが切り替わった時にLRUが動作しない可能性があり、キャッシュの使用率が低下するという問題がある。例えば、プロセスAにおいて、前記のプログラム510を実行し、スタックの最初の参照でLRUをロックした後で、LRUのロックが解除される前にプロセスAがプロセスBに切り替わった場合を考える。プロセスがBに切り替わっても、キャッシュの

30 LRUはロックされたままなので、ロックされたカラムにおける追い出し対象のブロックは、常にLRUがロックされたときに追い出し対象であったブロックとなり、プロセスBにおける該当カラムは、ダイレクトマップのキャッシュと同様の動作になってしまい、キャッシュの使用率が著しく低下する。またこれを避けるために、プロセスがAからBに切り替わる時、OSによってLRUのロックを全て解除することも考えられるが、この処理は大きなオーバーヘッドとなる。以上のように、従来方式では、LRUによって性能が低下する場合があり、かつ前記特許に示されているLRUのロック方式を使用した場合もマルチプロセス環境においてキャッシュの使用率が低下するという問題が発生する。本発明の目的は、前記のプログラム例510に示すような、空間的局所性はあるが、時間的局所性が無いデータの参照によって、時間的局所性のあるデータがキャッシュから追い出されることを減少させることができ、かつマルチプロセス環境においても特別な処理を行なうことなくLRUを正常に動作させることができる、キャッシュ記憶装置を提供することである。

【課題を解決するための手段】本発明は、ブロックの追

い出しアルゴリズムにLRUを使用するキャッシュ記憶装置において、キャッシュブロックのLRU化を指示できるメモリ参照命令を実行したとき、キャッシュ上の該当ブロックをLRU化することができる回路を備えることを特徴とする。従来、キャッシュブロックが参照された場合、該当するブロックのLRUは最新となり、最も追い出されにくくなるが、本発明は、該当ブロックの最後の参照時に上記LRU化命令を使用することで、アクセスしたキャッシュブロックを最も遠い昔に参照されたブロックにし（ブロックのLRU化）、次に同一カラムの追い出しが発生したときに、前記LRU化したブロックを最も追い出され易くすることを特徴とする。

【発明の実施の形態】以下、図面を参照して本発明の実施例を詳細に説明する。本実施例では、以下のキャッシュ記憶装置を例に説明する。

- (1) キャッシュのサイズは、128Kバイト
- (2) キャッシュのブロックサイズは128バイト
- (3) 4ウェイセットアソシアティブ
- (4) ブロック置き換えアルゴリズムにはLRUを使用
- (5) ストアスルーワイ

なお、本発明は、ブロック置き換えアルゴリズムにLRUを使用したセットアソシアティブ方式のキャッシュ記憶装置に適用するものあり、上記仮定におけるキャッシュサイズ、ブロックサイズ、ウェイ数が異なるキャッシュにも適用可能であり、またストアイン方式のキャッシュにも適用可能である。従来技術のセットアソシアティブ方式の説明において使用した図4は、本実施例で使用するキャッシュのメモリ部分の構造を示している。キャッシュメモリ部の動作は従来技術で説明した通りである。図1は、図4をメモリ部とする本発明のキャッシュ記憶装置の概略構成図である。101は図4におけるデータおよびアドレスアレイ406であり、111は図4におけるLRU407に対応する。本実施例のキャッシュ記憶装置は、キャッシュメモリが格納するデータおよびそのアドレスを保持するデータアレイおよびアドレスアレイ101と、参照履歴情報を保持するLRUアレイ111と、読み出したデータをセレクトするセレクタ108と、ヒット／ミスを判定するHIT／MISS判定回路122と、LRUアレイに書き込むLRUの値を作成するLRUエンコード回路110と、読み出したLRUの値から追い出すROWを決定するLRUデコード回路112と、データアレイおよびアドレスアレイから読み出したカラム内の4個のブロックのデータのうちの1つを選択するためのセレクト信号118を作成する選択ROW制御回路113と、データアレイ／アドレスアレイ／LRUアレイへの書き込みおよび読み出しを制御するDATA／TAG／LRU制御回路123から構成される。以下では、まず通常のロード命令を実行したときの、図1のキャッシュ記憶装置の動作、特にLRUデコード回路112とLRUエンコード回路110の動作に

について説明し、その後、本発明で提案するLRU化ロードを実行したときの、図1のキャッシュ記憶装置の動作、特にLRUエンコード回路110の動作について説明する。通常のロード命令を実行すると、命令が参照するアドレス102と、データおよびタグを更新するか否かを表すデータ／タグ更新信号106と、キャッシュからのデータの読み出しを指示するキャッシュ読み出し信号107が与えられる。上記信号が与えられると、まずアドレス102のカラムアドレス部によって、データアレイ／アドレスアレイがアクセスされ、該当カラム内のROW0、ROW1、ROW2、ROW3に格納されているデータおよびアドレスが、それぞれデータ0、1、2、3およびタグ0、1、2、3に読み出される。次にHIT／MISS判定回路122において、読み出された4個のタグの値とロード命令が参照するアドレスのうちのブロックアドレス部分を比較する。前記の4個の比較結果のうちのいずれかが一致していれば、ロード命令が参照するブロックがキャッシュ上に存在し、HIT／MISS判定回路122からのヒット信号116が1となる。いずれのROWとも一致しなければ、ロード命令が参照するブロックはキャッシュ上には存在せず、ヒット信号116は0となる。HIT／MISS判定回路122は、上記のヒット信号116とともに、ヒットしたROW番号を信号119に出力する。選択ROW制御回路113は、ヒット／ミスに応じて、読み出すべきROWを決定する。ここでヒットならば、ヒットROW119によって与えられるROW番号を選択ROW信号117に出力し、かつこれをセレクタ108のセレクト信号118に出力する。セレクタ108では、セレクト信号118を使用して、同時に読み出した4個のデータおよびタグから1つをセレクトし、キャッシュからの読み出しデータ114および読み出しタグ115をロード命令の結果として出力する。HIT／MISS判定回路122の結果がミスの場合、選択ROW制御回路113は、LRUデコード回路112から出力される追い出し対象ROWを選択ROW信号117に出力する。その後、主記憶からのデータのフェッチを行ない、フェッチした1ブロック分のデータをキャッシュに書き込む。ここでデータは、前述した、キャッシュミス時に選択ROW信号40117から出力された追い出し対象ROWに書き込む。前述のとおり、キャッシュミス時に追い出し対象となるROWは、LRUデコード回路112で決定される。このLRUデコード回路112の動作を図2および図3を用いて説明する。図2は、従来から使用されている4ウェイセットアソシアティブのLRUの実現方法を表している。この方法によると、4ウェイのLRUを6ビットで表し、かつ最も最近アクセスされたROWへのLRUの更新を、前記6ビットのうちの3ビットだけを用いて行なうことができる。本実施例におけるLRUの実現方法も、

この従来技術を使用する。図2の各ノード201、202、203、204は4ウェイセットアソシアティブにおける各ROWを表し、ノード間のエッジNEWER(0)～NEWER(5)はノード間の参照の新旧関係を表す。このエッジは有向エッジであり、矢印が指すノードは、矢印が出るノードよりも最近アクセスがあったROWであることを表す。したがって、4ノードのうちで最も遠い昔にアクセスされたノード(追い出し対象となるノード)は、どこからも矢印が指されていないノードとなる。図2に示す矢印の向きを初期状態とし、NEWER(0-5)=(0, 0, 0, 0, 0, 0)とする。この場合、初期状態で最も遠い昔にアクセスされたノードはROW0である。以上のように4ウェイのLRUを実現するため、1つのカラムに対して6ビットNEWERを使用する。本実施例ではNEWERは、図1のLRUアレイ111が保持する。キャッシュミスが発生すると、該当カラム内のROWのうちで、最も遠い昔にアクセスされたROWを追い出し対象のROWとして選択する。前述のように最も遠い昔に参照されたROWは、どこからも矢印が示されていないROWである。キャッシュミス時の追い出し対象のROWは、キャッシュアクセス時にLRUアレイから読み出される該当カラムのNEWER(0-5)をLRUデコード回路112でデコードして得られるROWを使用する。図3の表301はLRUデコード回路112の制御を表している。表301はNEWER(0-5)が与えられたときに、最も遠い昔に参照されたROWを検出する論理である。例えば、ROW0が最も遠い昔にアクセスされたROWとなるのは、ROW0とつながる矢印NEWER(0)、NEWER(3)、NEWER(4)が全てROW0から出ている場合である。すなわち各矢印が初期状態と同じNEWER(0, 3, 4)=(0, 0, 0)の場合である。同様にROW1が追い出し対象となるのは、NEWER(0)だけが初期状態と異なる、NEWER(0, 1, 5)=(1, 0, 0)の場合である。以上が、通常のロード命令を実行したときの、LRUによる追い出しROWの決定方法(LRUデコード回路110の動作)である。次に通常ロード命令を実行したときの、LRUの更新動作(LRUエンコード回路112の動作)について説明する。通常のロード命令の実行では、以下の場合にLRUを更新する。

(1) キャッシュヒット時

(2) キャッシュミス時の主記憶からのデータのキャッシュへの書き込み時

(1) (2) のケースは共に該当ROWのLRUを最新(最も最近参照されたROW)に変更する。(1)のケースでは、キャッシュにヒットすると、HIT/MIS S判定回路122によって出力されるヒットROW119が、選択ROW制御回路113で選択され、選択ROW信号117に出力される。その後、選択ROW信号11

7はLRUエンコード回路110に入力され、LRUエンコード回路110において、ROW番号がLRUアレイに登録するNEWER(0-5)にエンコードされ、LRUアレイが更新される。通常のロード命令の実行において、LRUエンコード回路110は、図3の表302にしたがって動作する。例えば、LRUエンコード回路110への入力がROW0である場合、ROW0が最新になるように、該当カラムのNEWER(0-5)を変更する。表302によると、このケースではNEWER(0, 3, 4)=(1, 1, 1,)に変更する。これは図2におけるROW0に繋がる矢印が全てROW0を向くように変更することを意味する。同様にLRUエンコード回路110への入力がROW1は、ROW1を最も最近参照したROWに変更するため、LRUアレイの該当カラムにNEWER(0, 1, 5)=(0, 1, 1)を登録する。次に(2)のキャッシュへの書き込みケースについて説明する。まずキャッシュミスが生じると、LRUアレイから該当カラムのNEWER(0-5)が送出される。これがLRUデコード回路112に入力され、最も遠い昔に参照されたROWが追い出し対象として出力される。LRUデコード回路112の処理は、前述した図3の表301にしたがって行なわれる。この追い出し対象のROWは選択ROW制御回路113に入力され、選択ROW信号として出力される。その後、主記憶からデータが戻り、キャッシュに書き込むときに、(1)のケースと同様に、前記の追い出し対象ROW番号が、LRUエンコード回路110に入力され、データを書き込むと同時に該当ROWが最も最近参照されたROWとなるように該当カラムのLRUアレイを変更する。ここでLRUエンコード回路110における処理は、(1)のケースと同様に、図3の表302に基づいて動作する。以上が通常のロード命令を実行したときのキャッシュ記憶装置の動作である。次に本発明で開示するLRU化ロード命令を実行した時のキャッシュメモリの動作について説明する。上述したように、通常のロード命令を実行すると、参照したブロックのLRUは最新になるのに対して、LRU化ロード命令を実行すると、参照したブロックのLRUは、最も遠い昔に参照したブロックになる。これにより、後続の命令で同一カラムを参照し、かつキャッシュミスが発生した場合は、LRU化ロード命令で参照され、LRU化されたROWが追い出し対象になる。LRU化ロード命令が実行されると、通常のロード命令の場合と同様に、参照するアドレスのカラム部によってデータアレイ、アドレスアレイ、LRUアレイがアクセスされ、該当するカラムのデータ、タグ、LRUが読み出される。ここでヒットすれば、ヒットしたROWのデータとタグが読み出しデータ114および読み出しタグ115によって返される。次にLRU化ロード命令の実行によるLRUアレイの更新について説明する。LRUの更新は通常ロードと同様

に、以下のケースで発生する。

(1) キャッシュヒット時

(2) キャッシュミス時の主記憶からのデータのキャッシュへの書き込み時

LRU化ロード命令を実行するときは、(1) (2) のケースは共に該当ROWのLRUを最も遠い昔に参照されたROWに変更する。(1)のヒット時には、HIT/MISS判定回路122からヒットしたROW119が出力され、選択WAY信号117に、119で示されるROWが出力される。その後、選択WAY信号117はLRUエンコード回路110に入力され、LRUアレイ111に書き込むべきNEWER(0-5)の値を決定する。LRU化ロード時は、LRUエンコード回路110は図3の表303にしたがって動作する。例えば、LRU化ロード命令がROW0にヒットした場合、LRUエンコード回路110に、ROW0が入力される。表303によると、ROW0が入力されると、NEWER(0, 3, 4) = (0, 0, 0)が出力される。これは図2におけるノードROW0に繋がっている矢印が、通常のロード命令とは逆に、全てROW0から出るようNEWERを変更することを意味しており、これによりROW0は該当カラム内で最も遠い昔に参照されたROWに変更される。同様にLRU化ロード命令がROW1にヒットした場合、LRUエンコード回路110にはROW1が入力され、NEWER(0, 1, 5) = (1, 0, 0)が出力される。これは図2におけるノードROW1に繋がっている矢印が、通常のロード命令とは逆に、全てROW1から出るようNEWERを変更することを意味しており、これによりROW1は該当カラム内で最も遠い昔に参照されたROWに変更される。次に(2)のキャッシュへの書き込み時の動作について説明する。LRU化ロード命令によるキャッシュアクセスでミスが発生すると、HIT/MISS判定回路122からヒット信号116によってミスが発生したことが選択ROW制御回路113に伝えられる。またLRUデコード回路112からは、LRU化ロード命令がアクセスしたカラム内の最も遠い昔に参照されたROWが作成され、選択ROW制御回路113に入力される。選択ROW制御回路はキャッシュミスが発生すると、選択ROW信号117がLRUデコード回路113から入力されるROW番号を、追い出し対象ROWとして出力する。LRU化ロード命令のキャッシュミスによって主記憶からデータがフェッチされると、そのデータをキャッシュに書き込むと同時にLRUの更新も行なう。ここで新しいLRUの値は、LRU化ロード命令のミス時に追い出しROWとして選択ROW信号117から出力されたROW番号であり、この値はLRUエンコード回路110に入力され、LRUアレイ111への入力信号が作成される。なお、LRU化ロード命令による書き込み時の、LRUエンコード回路110は、前述したLRU化ロード命令によってデータが書き込まれたブロックのLRUは、最も遠い昔にアクセスされたブロックとなる。LRU化専用命令を実行すると、まず指定されたアドレスでキャッシュのヒットチェックを行なう。キャッシュミスならば処理を終了する。キャッシュヒットならば、ヒットしたROWが選択ROW信号117から出力

命令のヒットケースと同様に、図3の表303にしたがって動作する。すなわち、LRUエンコード回路110に入力された書き込みROWのLRUが、最も遠い昔にアクセスしたROWとなるように、LRUアレイへの入力信号NEWER(0-5)が作成される。上記実施例では、通常のロード命令の機能と、該当キャッシュブロックをLRU化する機能を備えるLRU化ロード命令について説明したが、請求項で示した以下の命令についても同様に実施することができる。

・ LRU化ストア命令

通常のストア命令の機能と、該当キャッシュブロックをLRU化する機能備えた命令。

・ LRU化専用命令

命令で指定したアドレスのブロックがキャッシュ上にあれば、そのブロックをLRU化するための専用命令。

・ LRU化プリフェッチ命令

通常のプリフェッチ命令の機能と、プリフェッチしたキャッシュブロックをLRU化する機能を備えた命令。

・ LRU化オフセット指定付きプリフェッチ命令

20 通常のプリフェッチ命令の機能と、プリフェッチアドレスから、命令で指定されたオフセットだけ離れたアドレスのブロックがキャッシュ上にあれば、このブロックをLRU化する機能を備えた命令。

・ LRU化オフセット指定付きメモリ参照命令

ロード命令、ストア命令といった通常のメモリ参照命令の機能と、メモリ参照するアドレスから、命令で指定されたオフセットだけ離れたアドレスのブロックがキャッシュ上にあれば、このブロックをLRU化する機能を備えた命令。以下では、これらの命令を実行したときの、

30 本実施例のキャッシュ記憶装置の動作について説明する。LRU化ストア命令を実行すると、まず与えられたアドレスで、キャッシュをアクセスし、ヒットチェックを行なう。キャッシュミスならば、本実施例はストアスルーバイパスしているので、主記憶にデータを書き込んで処理を終了する。キャッシュヒットの場合は、ヒットしたROWが選択ROW信号117から出力される。その後、ストアデータをキャッシュに書き込む時に、前記選択ROW信号117がLRUエンコード回路110に入力され、LRUエンコード回路110によって、LRU

40 アレイ111に書き込むべき値が作成され、キャッシュのデータを更新すると共にLRUの値も更新される。なお、ここでLRUエンコード回路110は、前述したLRU化ロード命令におけるLRUの更新時と同様に、図3の表303にしたがって動作する。これにより、LRU化ストア命令によってデータが書き込まれたブロックのLRUは、最も遠い昔にアクセスされたブロックとなる。LRU化専用命令を実行すると、まず指定されたアドレスでキャッシュのヒットチェックを行なう。キャッシュミスならば処理を終了する。キャッシュヒットならば、ヒットしたROWが選択ROW信号117から出力

され、LRUエンコード回路110に入力される。LRUエンコード回路110では、LRU化ロード命令におけるLRUの更新時と同様に、図3の表303にしたがって、更新するLRUの値を生成し、LRUアレイ111に書き込む。これにより、LRU化専用命令によってアクセスされたキャッシュブロックは最も遠い昔にアクセスされたブロックとなる。LRU化プリフェッチ命令を実行すると、まずプリフェッチアドレスでキャッシュをアクセスし、ヒットチェックを行なう。キャッシュミスならば、主記憶からのデータのフェッチを行なう。ここでフェッチしたデータを書き込むキャッシュのROWは、選択ROW信号117から出力される。このROW番号は、LRU化ロード命令におけるキャッシュミス時と同様にLRUデコード回路112によって作成される。フェッチしたデータを書き込む時に、前記の書き込むべきROW番号をLRUエンコード回路110に入力し、更新するLRUの値を決定する。LRUエンコード回路110は、LRU化ロード命令におけるLRUの更新時と同様に、図3の表303にしたがって動作する。これによりキャッシュに書き込んだブロックは、最も遠い昔にアクセスされたブロックとなる。なお、プリフェッチ命令では、主記憶からフェッチしたデータのレジスタへの転送は行なわない。次にLRU化プリフェッチ命令でキャッシュヒットした場合は、LRU化専用命令でキャッシュヒットした場合と同様に動作する。アドレスアレイが2ポートのキャッシュにおいて、LRU化オフセット指定付きプリフェッチ命令を実行すると、まず指定されたプリフェッチアドレスと、前記プリフェッチアドレスから命令で指定されたオフセット分離されたアドレス（以後、“LRU化アドレス”と呼ぶ）によってキャッシュをアクセスし、ヒットチェックを前記の2個のアドレスで同時に行なう。プリフェッチアドレスによるキャッシュアクセスは、通常のプリフェッチ命令の動作と同様に行なう。すなわちヒットしていれば、データのレジスタへの転送は行なわず、ミスしていれば主記憶からデータをフェッチするが、レジスタへのデータの転送は行なわない。なお、プリフェッチアドレスによるキャッシュアクセスでは、アクセスしたキャッシュブロックのLRUは、従来のプリフェッチ命令と同様に最新のブロックに変更する。一方LRU化アドレスによってアクセスされたキャッシュブロックはLRU化する。このLRU化の動作は、LRU化専用命令の動作と同様である。これにより、プリフェッチしたデータを保持するキャッシュブロックのLRUは最新となり、プリフェッチアドレスとオフセットによって生成したLRU化アドレスを保持するキャッシュブロックのLRUは、最も遠い昔に参照されたブロックとなる。最後に、LRU化オフセット指定付きメモリ参照命令を、アドレスアレイが2ポートのキャッシュで実行した場合を説明する。前記命令を実行すると、まずメモリ参照するアドレスと、前記メモ

リ参照アドレスから命令で指定されたオフセット分離されたアドレス（LRU化アドレス）によってキャッシュをアクセスし、ヒットチェックを前記の2個のアドレスで同時に行なう。メモリ参照アドレスによるキャッシュアクセスは、通常のメモリ参照命令と同様に行なう。すなわち、該当するキャッシュブロックのLRUは最も最近参照されたブロックとなるように変更する。一方LRU化アドレスによってアクセスされたキャッシュブロックはLRU化する。LRU化の動作は前述したLRU化オフセット指定付きプリフェッチ命令におけるLRU化アドレスによるキャッシュアクセスと同様である。なお上記のLRU化オフセット指定付きプリフェッチ命令とLRU化オフセット指定付きメモリ参照命令の実施例では、LRU化アドレスを、プリフェッチアドレスおよびメモリ参照アドレスからオフセット分離されたアドレスとしたが、オフセット×ブロックサイズ分離されたアドレスとする等、オフセットを用いてLRU化アドレスをエンコードすることも可能である。また、上記実施例では、2ポートのアドレスアレイを用いて、プリフェッチアドレスおよびメモリ参照アドレスと、LRU化アドレスで同時にキャッシュをアクセスしたが、1ポートのキャッシュで実施する場合は、2度キャッシュをアクセスすることで同様の動作を実現することができる。以上、本発明で提供するキャッシュ記憶装置において、各LRU化命令を実行したときの、キャッシュ記憶装置の動作について説明した。以下では、従来技術で問題が発生したプログラム例501に、LRU化命令を適用することで、前記の問題が解決できることを示す。なお、プログラム例501の動作および仮定は、従来技術の説明と同じである。図5の520は、510にLRU化ロード命令およびLRU化ストア命令を適用した、変換後のプログラムである。520では、LRU化命令の適用のために、まず内側ループ513を16回展開して内側ループ523に変換する。これにより、内側ループは、1回の繰り返してキャッシュ1ブロック分の128バイトのデータを参照するコードになる。次にキャッシュブロックの最後の参照命令である525と526をLRU化命令に変換する。520を実行すると、内側ループのLRU化命令を実行したとき、従来は該当ブロックのLRUが最新になるとところが、LRUが最も遠い昔に参照された状態になり、次に同一カラムへのアクセスが発生した時には、LRU化したブロックが追い出し対象になる。これにより、配列Sはキャッシュの各カラムの1つのROWを使い回すことになり、内側ループが終了したときも、局所変数が配列Sによって追い出されず、局所変数の使用522でキャッシュにヒットする。

【発明の効果】本発明によれば、命令セットにLRU化命令を設け、かつこのLRU化命令が実行されたとき、該当するキャッシュブロックのLRU情報を、最も遠い昔に参照されたブロック変更する回路をキャッシュ記憶

装置に具備することにより、空間的局所性はあるが、時間的局所性のないデータの参照によって、時間的局所性のあるデータがキャッシュからおいたされることを抑止することができる。またLRU化命令は、将来参照されないブロックを追い出し対象のブロックに変更する命令であり、LRUの変更をロックする方式のように、ロックを解除する必要がないため、プロセスが切り替わってもLRUは動作し続け、キャッシュの使用効率を低下させることはない。

(図面の簡単な説明)

【図1】本発明の一実施例であるキャッシュ記憶装置の概略構成図である。

【図2】図1に示すキャッシュ記憶装置におけるLRU方式の追い出しアルゴリズムの実現方法を示す図である。

【図3】図1に示すLRUエンコード回路およびLRUデコード回路の制御方法を示す図である。

【図4】従来例および本発明のセットアソシティブキヤッシュのメモリ部の概略構成を示す図である。

【図5】従来例で性能が低下する場合のプログラム例お*20

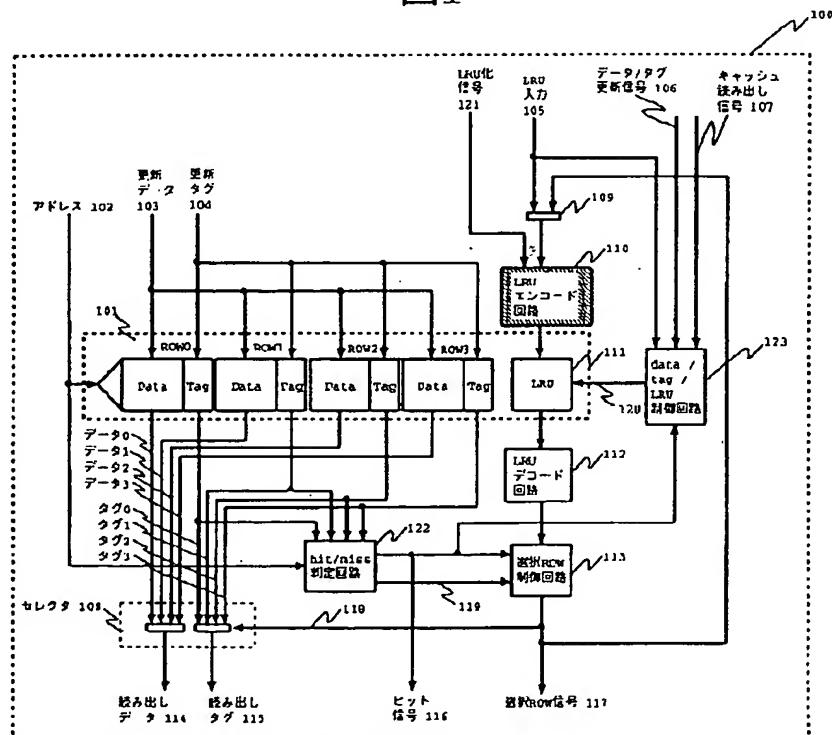
* よび、このプログラム例に対して本発明を適用したときに生成されるプログラム例を示す図である。

【符号の説明】

100…キャッシュ記憶装置、101…メモリ部、102…キャッシュを参照するアドレス、103…キャッシュを更新するデータ、104…キャッシュを更新するタグ、105…更新するLRUの入力、106…データおよびタグを更新することを伝える信号、107…キャッシュの読み出しを指示する信号、108…各ROWから読み出したデータのセレクタ、109…LRU更新時のLRUのセレクタ、110…LRUエンコード回路、111…LRUメモリ部、112…最も遠い昔に参照されたROWを検出するLRUデコード回路、113…読み出したLRUのセレクト回路、114…読み出しデータ、115…読み出しタグ、116…ヒット信号、117…選択ROW制御回路、119…HITRO制御回路、120…データ/タグ/LRU制御信号、122…ヒット/ミス判定回路、123…データ/タグ/LRUの制御回路。

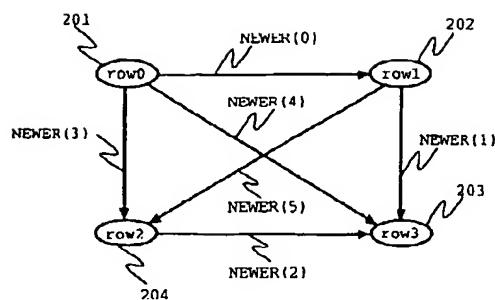
[図1]

1



【図2】

図2



【図3】

図3

図3に示すように、リプレースrowの決定:デコード表、通常のメモリ参照によるLRU更新:テンコード表、LRU化メモリ参照によるLRU更新:エンコード表が示されています。

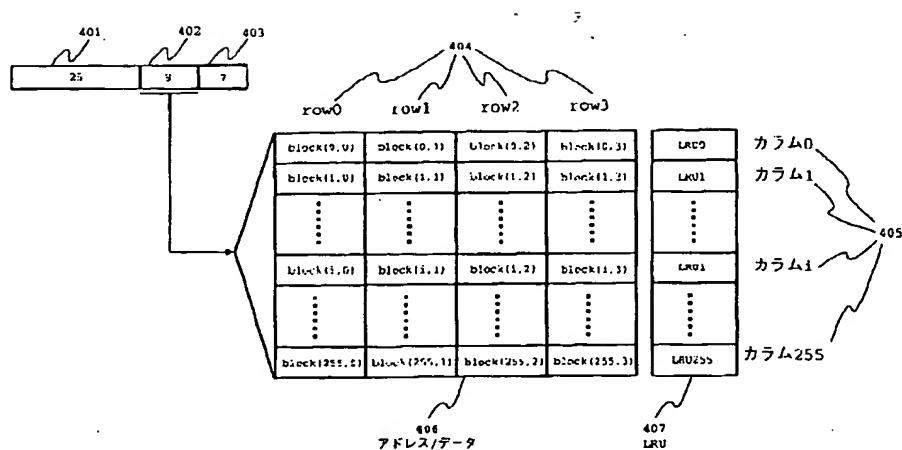
row	(0)	(1)	(2)	(3)	(4)	(5)
row0	0		0	0		
row1	1	0				0
row2		0	1		1	
row3	1	1		1		

row	(0)	(1)	(2)	(3)	(4)	(5)
row0	1			1	1	
row1	0	1				1
row2		1	0		0	
row3	0	0		0		

row	(0)	(1)	(2)	(3)	(4)	(5)
row0	0		0	0		
row1	1	0				0
row2		0	1		1	
row3	1	1		1		

【図4】

図4



[図5]

図5

[プログラム例]

```

#define N 100
#define M 16000
float S[N][M];
float T[N][M];

foo () {
    int a,b,c,d;
    int i,j;

    for ( i = 0; i < N; i ++ ) {
        ;
        for ( j = 0; j < M; j ++ ) {
            T[i][j] = S[i][j];
        }
    }
}

```

501 502 503 504 505 506 507 508 509 510

[疑似命令列]

```

#define N 100
#define M 16000
float S[N][M];
float T[N][M];

foo () {
    int a,b,c,d;
    int i,j;

    for ( i = 0; i < N; i ++ ) {
        ;
        for ( j = 0; j < M; j ++ ) {
            LD r1 S[i][j];
            ST r1 T[i][j];
        }
    }
}

```

511 512 513 514 515 516 517 518 519 520

[LRU化ロード命令の使用]

```

#define N 100
#define M 16000
float S[N][M];
float T[N][M];

foo () {
    int a,b,c,d;
    int i,j;

    for ( i = 0; i < N; i ++ ) {
        ;
        for ( j = 0; j < M; j = j + 16 ) {
            LD r1 S[i][j];
            ST r1 T[i][j];
            LD r1 S[i][j+1];
            ST r1 T[i][j+1];
            LD r1 S[i][j+2];
            ST r1 T[i][j+2];
            LD r1 S[i][j+3];
            ST r1 T[i][j+3];
            LD r1 S[i][j+4];
            ST r1 T[i][j+4];
            LD r1 S[i][j+5];
            ST r1 T[i][j+5];
            LD r1 S[i][j+6];
            ST r1 T[i][j+6];
            LD r1 S[i][j+7];
            ST r1 T[i][j+7];
            LD r1 S[i][j+8];
            ST r1 T[i][j+8];
            LD r1 S[i][j+9];
            ST r1 T[i][j+9];
            LD r1 S[i][j+10];
            ST r1 T[i][j+10];
            LD r1 S[i][j+11];
            ST r1 T[i][j+11];
            LD r1 S[i][j+12];
            ST r1 T[i][j+12];
            LD r1 S[i][j+13];
            ST r1 T[i][j+13];
            LD r1 S[i][j+14];
            ST r1 LRU r1 S[i][j+14];
            LD r1 LRU r1 T[i][j+14];
        }
    }
}

```

521 522 523 524 525 526

16回展開

LRU化ロード、ストア命令
1ライン(128バイト)中の
最後の8バイトの参照

フロントページの続き

(72)発明者 釜田 栄樹

神奈川県秦野市堀山下1番地 株式会社日
立製作所エンタープライズサーバ事業部内

F ターム(参考) 58005 JJ13 MM01 QQ02 TT02